Programación Orientada a Objetos (POO)

La POO es una técnica para desarrollar soluciones computacionales utilizando componentes de software (objetos de software). La POO se basa en el modelo objeto donde el elemento principal es el objeto, el cual es una unidad que contiene todas sus características y comportamientos en sí misma, lo cual lo hace como un todo independiente pero que se interrelaciona con objetos de su misma clase o de otras clase, como sucede en el mundo real.

Objeto: Componente o código de software que contiene en sí mismo tanto sus características (Propiedades) como sus comportamientos (métodos); se accede a través de su interfaz o signatura.

Propiedad o Campo: Es una característica de un objeto, que ayuda a definir su estructura y permite diferenciarlo de otros objetos. Se define con un identificador y un tipo, el cual indica los valores que puede almacenar. El conjunto de valores de los campos definen el estado del objeto.

Método: Es la implementación de un algoritmo que representa una operación o función que un objeto realiza. El conjunto de los métodos de un objeto determinan el comportamiento del objeto.

Según lo anterior responda las siguientes preguntas:

1. Cuál de las siguientes Palabras podrían definir a un Objeto:

A. Correr
B. Saltar
C. Ventana
D. Color

2. Cuál de las siguientes Palabras define mejor una Propiedad:

A. Click
B. Cargar
C. Arrastrar
D. Color de Fondo

3. Cuál de las siguientes palabras define mejor un Método:

A. Borde
B. Ancho
C. Tirar
D. Velocidad

Un evento es una señal que comunica a una aplicación que ha sucedido algo importante. Por ejemplo, cuando un usuario hace clic en un control de un formulario, el formulario puede provocar un evento Click y llamar a un procedimiento que controla el evento. Los eventos también permiten que las tareas separadas se comuniquen. Suponga, por ejemplo, que una aplicación realiza una tarea de escritura en disco independientemente de la aplicación

principal. Si un usuario cancela la escritura en disco, la aplicación puede enviar un evento de cancelación que ordene la detención del proceso de ordenación.

4. Cuál de los siguientes NO es un evento:

A. Click
B. Color
C. Doble Click
D. Movimiento De Mouse

En programación, una variable está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Ese espacio contiene una cantidad o información conocida o desconocida, es decir un valor. El nombre de la variable es la forma usual de referirse al valor almacenado: esta separación entre nombre y contenido permite que el nombre sea usado independientemente de la información exacta que representa.

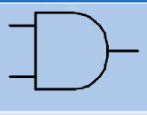
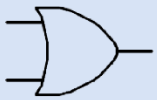
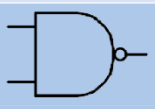
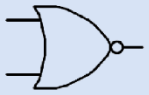
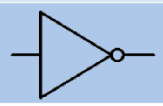
5. Según lo anterior cuál de estas palabras serían un buen identificador para una variable que almacene el valor del temperatura de un sensor:

A. sueldo
B. 1000000
C. Calcular
D. temperatura

6. Por qué son útiles las variables:

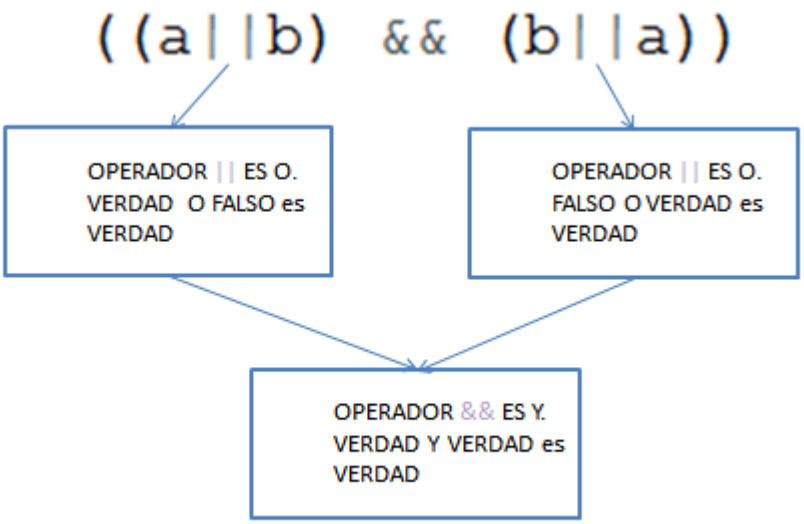
A. Permiten Identificar un valor y hacer operaciones
B. Permiten almacenar información y utilizarla en varias partes del programa
C. Permiten cambiar las propiedades de un objeto
D. Permite efectuar acciones sobre los programas según la forma en que se utilicen

Un operador es un elemento de programa que se aplica a uno o varios operandos en una expresión o instrucción. Un operador, es un símbolo que indica al compilador que se lleve a cabo ciertas manipulaciones matemáticas o lógicas.

Operador	*Nomenclatura en Arduino*	*Significado*	*Símbolo*
AND	&&	Devuelve ***true*** cuando la primera **Y** la segunda condición se cumplen	
OR	||	Devuelve ***true*** cuando la primera **O** la segunda condición se cumple	
NAND	(¡=) && (¡=)	Devuelve ***true*** cuando **NI** la primera **NI** la segunda condición se cumplen	
NOR	(¡=) || (¡=)	Devuelve ***true*** cuando **NO** se cumple alguna de las condiciones	
NOT	¡=	Devuelve ***true*** cuando **NO** se cumple una condición	

Los operadores lógicos permiten hacer comparaciones lógicas como "Si ocurre una situación y también otra situación, el resultado va a ser una respuesta positiva". Eso es lógica… veamos un ejemplo con un led de Arduino que va a recibir un VERDADERO :

```
bool a = FALSE;     // FALSE es FALSO o 0
bool b = TRUE;      // TRUE es VERDAD o 1
if ((a||b) && (b||a)){
  digitalWrite(LED_BUILDING, TRUE);
}
Else {
  digitalWrite(LED_BUILDING, FALSE);
}
```



Analizando el código anterior y observando los valores contenidos en las variables a y b.

7- La condicional if recibe como respuesta a la condición:

a- (TRUE)
b- (FALSE)
c- (FALSO)
d- NO RECIBE NINGUN VALOR LÓGICO

8- Si cambiamos los valores iniciales de las variables a y b por FALSE Y FALSE, respectivamente. La condición if recibe como condición.

a- (1)
b- (0)
c- (TRUE)
d- (TRUST)

9- Si cambiamos los valores iniciales de las variables a y b por TRUE Y TRUE, respectivamente. La condición if recibe como condición.

a- (0)
b- (FALSE)
c- (1)
d- (ERROR)

```
int tiempo=500
```

```
void loop() {
  if (tiempo < 300){
    digitalWrite(LED_BUILTIN, LOW);
  }
  else{
    digitalWrite(LED_BUILTIN, HIGH);
    delay(tiempo);
    digitalWrite(LED_BUILTIN, LOW);
    delay(tiempo);
    tiempo=tiempo-200;
  }
}
```

PRUEBA DE ESCRITORIO

# LOOP	VALOR DE VARIABLE
0	tiempo = 500
1	tiempo = 300
2	tiempo = 100
3	tiempo = -100

Observa el programa y analiza la Prueba de escritorio, la cual permite ver ciclo por ciclo como va cambiando el valor de la variable. De acuerdo a lo anterior responde:

10 El led se prende

a- Ninguna vez
b- 1 vez
c- Dos veces
d- Tres veces

11 La primer vez que se prende el led dura prendido

a- 500 ms
b- 300 ms
c- 100 ms
d- -100 ms

Analiza el siguiente algoritmo al cual escribe los números del 1 al 50, se le han numerado las instrucciones y responde

```
1.  Algoritmo numerosnaturales_1a50
2.  Num<-1
3.  Resul<-0
4.  Repetir
5.          Resul<-Num
6.          Num<-Num+1
7.  Hasta Que Num>50
8.  Escribir Resul
9.  Fin algoritmo
```

12. Que línea cambiaría para que la lista fuera de solo números pares.

a. La dos y la seis

b. La tres y la siete

c. La dos o la 7

d. La 3 o la 8

13. Que línea cambiaría para que mostrara los números de 1 a 100

a. La 3

b. La 6

c La 1

d La 7

14. Instrucción de asignación Num <- Num + 1, es llamada incrementador porque:

a. Aumenta el tamaño de la memoria

b. Aumenta los resultados del algoritmo

c. Mejora el rendimiento de la máquina

d Suma uno a la variable cada vez que se ejecuta

15 La instrucción *Repetir* es un bucle que se repite hasta que

a. Se acaba el tiempo de ejecución

b. Se cumple la condición Num es mayor a 50

c. Se cumple la condición Num es menor a 50

d. Se cumple el ciclo predeterminado en Num

Analizando el siguiente programa

```
bool Prendido = 0;
int Tiempo = 1000;


loop(){
  while (Tiempo >= 1000 && Tiempo <=200){
    digitalWrite(LED_BUILDING, Prendido);
    delay(Tiempo);
    Tiempo = Tiempo - 100;
    Prendido = !Prendido;
  }
}
```

1- ¿De acuerdo al código anterior, cuanto tiempo en milisegundos el LED interno queda prendido después de 2 segundos?

a- Verdadero
b- Falso
c- No se puede determinar
d- Depende del Tiempo

Teniendo las siguientes líneas de código

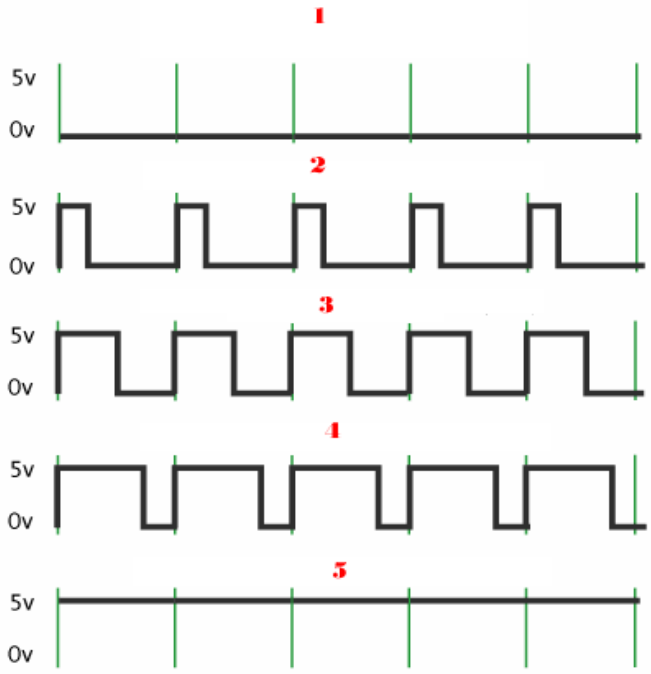
```
char Pepito[];
char Juanito[];
.
.
.

loop(){
        if (Juanito[]=="inquieto" || Pepito[]=="juicioso"){
                digitalWrite(LED_BUILTIN,0);
        }
        else{
                digitalWrite(LED_BUILTIN,1);
        }
```

2- El led se enciende si:

a- Pepito[] ="desjuiciado"; Juanito[] ="inquieto";
b- Pepito[] ="juicioso"; Juanito[] ="quieto";
c- Pepito[] ="juicioso"; Juanito[] ="inquieto";
d- Pepito[] ="desjuiciado"; Juanito[] ="quieto";

La modulación por ancho de pulso, o PWM, es una técnica para obtener resultados analógicos con medios digitales. El control digital se utiliza para crear una onda cuadrada, una señal activada y desactivada. Este patrón de encendido y apagado puede simular voltajes entre encendido total (5 voltios) y apagado (0 voltios) cambiando la parte del tiempo que la señal pasa en comparación con el tiempo que la señal pasa. La duración de "on time" se denomina ancho de pulso. Para obtener valores analógicos variables, cambia o modula ese ancho de pulso. Si repite este patrón de encendido / apagado lo suficientemente rápido con un LED, por ejemplo, el resultado es como si la señal fuera un voltaje constante entre 0 y 5v que controla el brillo del LED.

En el siguiente gráfico, las líneas verdes representan un período de tiempo regular. Esta duración o período es el inverso de la frecuencia PWM. En otras palabras, con la frecuencia PWM de Arduino a aproximadamente 500Hz, las líneas verdes medirían 2 milisegundos cada una. Una llamada a analogWrite () está en una escala de 0 a 255, de modo que analogWrite (255) solicita un ciclo de trabajo del 100% (siempre activado), y analogWrite (127) es un ciclo de trabajo del 50% (en la mitad del tiempo) para ejemplo.



3- De acuerdo a lo expuesto anteriormente, cual grafica representa la onda generada por la instrucción analogWrite(192).

a- 1
b- 2
c- 3
d- 4

4- De acuerdo a lo expuesto anteriormente, cual grafica representa la onda generada por la instrucción analogWrite(62).

a- 5
b- 4
c- 3
d- 2

La definición de la función random es

random(min, max)

5- Esto significa que:

a- min y max son los parámetros que definen el rango en el cual se desea el valor de retorno.
b- Es una función que debe tener dos variables

c- Es una variable aleatoria
d- Es una función para sacar un numero de 1 a 256